



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА - Российский технологический университет"

РТУ МИРЭА

**Институт Информационных Технологий
Кафедра Вычислительной Техники**

**ПРАКТИЧЕСКАЯ РАБОТА
по дисциплине
«Системный анализ данных СППР»**

Студент группы:ИКБО-42-23

Голев С.С.
(*Ф. И. О. студента*)

Преподаватель

Железняк Л.М.
(*Ф.И.О. преподавателя*)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
5 ПЧЕЛИНЫЙ АЛГОРИТМ	4
5.1 Цель и задачи практической работы	4
5.2 Постановка задачи	4
5.3 Ручной расчёт	5
5.4 Результат работы.....	5
5.5 Результат работы нахождения кратчайшего пути	6
ЗАКЛЮЧЕНИЕ	7
СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ	8
ПРИЛОЖЕНИЯ.....	9

ВВЕДЕНИЕ

Пчелиный алгоритм относится к классу роевых методов оптимизации, основанных на моделировании поведения пчёл при поиске источников пищи. Основная идея заключается в разделении особей на разведчиков и рабочих пчёл, которые исследуют пространство решений и обмениваются информацией о найденных участках с высокой «пригодностью». Разведчики отвечают за поиск новых областей, тогда как рабочие пчёлы сосредотачиваются на улучшении уже найденных решений. Такой подход обеспечивает баланс между глобальным поиском и локальным уточнением. В работе рассматривается применение пчелиного алгоритма для нахождения минимума функции. Благодаря кооперативному поведению и механизму обмена информацией пчелиный алгоритм демонстрирует высокую эффективность и способность избегать попадания в локальные минимумы.

5 ПЧЕЛИНЫЙ АЛГОРИТМ

Алгоритм основан на поведении роя пчёл, и представляет из себя стаю разведчиков, которые изначально находят лучшие точки, и потом в окрестности этих точек посылаются рабочие пчёлы которые продолжают искать точки экстремума.

5.1 Цель и задачи практической работы

Целью практической работы является изучение принципов пчелиного алгоритма и его применение для решения задач нахождения точек экстремума.

Для достижения поставленной цели необходимо выполнить следующие задачи:

1. Выполнить ручной расчёт одной итерации пчелиного алгоритма для функции Била;
2. Реализовать пчелиный алгоритм на языке Python для автоматического поиска точек экстремума;
3. Применить алгоритм для функции Била;

Таблица 5.1 – Функция Била

Формула	Глобальный минимум	Метод поиска
$f(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$	$f(3, 0.5) = 0$	$-4.5 \leq x, y \leq 4.5$

5.2 Постановка задачи

В рамках практической работы необходимо реализовать пчелиный алгоритм вручную и кодово, алгоритм будет проверяться на функции Била.

5.3 Ручной расчёт

Выполним ручной расчёт одной итерации данного алгоритма на примере функции Била.

Первоначально на случайные точки отправляются пчёлы разведчики. Из всех точек, на которые попали пчёл разведчики выбирается n лучших точек и m перспективных точек.

Рассмотрим пчелу, которая попала на координаты $(2, 1)$, значения функции в данной точке будет равно 14.203125 . Для дальнейшего расчёта будем воспринимать эту точку как лучшее значение в рое.

Теперь в область этой точки отправим рой рабочих пчёл.

Для первой координаты $[2 - 1 = 1; 2 + 1 = 3]$

Для второй координаты $[1 - 1 = 0; 1 + 1 = 2]$

В эту область отправляем рабочих пчёл, и рассчитываем значение функции в точках рабочих пчёл, данные действия производятся для всех найденных лучших и перспективных областей, после чего среди всех новых точек снова отмечаются лучшие, а процесс повторяется.

5.4 Результат работы

Реализуем нахождение точек экстремума с помощью пчелиного алгоритма, количество пчёл разведчиков: 10, пчёл отправляющихся в лучшие области 5, лучших областей 2 количество итераций: 1000, выполним реализацию на языке Python. Реализация представлена в приложении Д.

```
(venv) PS C:\Users\semen\Desktop\MIREA\System_data_analysis\Practice5> py .\pchela.py
14.203125
Эталон: f(3, 0.5) = 0
Результат x = 3.01150 y = 0.50096 f = 0.00010
```

Рисунок 5.1 – Пример выполнения пчелиного алгоритма

5.5 Результат работы нахождения кратчайшего пути

В ходе практической работы был выполнен ручной расчёт одной итерации пчелиного алгоритма.

Данный расчёт демонстрирует работу пчелиного алгоритма: пчелы разведчики находят лучшие и перспективные точки, на которые отправляются рабочие пчёлы и происходит поиск экстремума функции.

Далее была выполнена кодовая реализация данного алгоритма, которая позволила автоматически находить точки минимума функции;

В результате работы показано, что пчелиный алгоритм является эффективным методом для точек экстремума функции и может использоваться в задачах оптимизации.

ЗАКЛЮЧЕНИЕ

В ходе работы был реализован пчелиный алгоритм для решения задач оптимизации различного типа. Проведённые эксперименты показали, что данный метод эффективно приближается к глобальному минимуму. Особенностью алгоритма является сочетание глобального поиска, осуществляемого разведчиками, и локального уточнения, выполняемого рабочими пчёлами. Благодаря механизму обмена информацией между агентами обеспечивается адаптация к структуре задачи и устойчивость к случайным возмущениям. Полученные результаты подтвердили эффективность и надёжность пчелиного алгоритма, а также его применимость для решения широкого круга сложных оптимизационных задач.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Python Software Foundation. Python Documentation — [Электронный ресурс]. URL: <https://docs.python.org/3/> (дата обращения: 15.09.2025).
2. Лутц М. Изучаем Python. 5-е изд. / пер. с англ. — Санкт-Петербург: Символ-Плюс, 2019. — 1648 с.
3. Балляев С. А. Объектно-ориентированное программирование. Учебное пособие. — Москва : ФОРУМ, ИНФРА-М, 2020. — 256 с.
4. Гринберг Д. Программирование на Python 3. Подробное руководство. — Москва : Вильямс, 2014. — 832 с.

ПРИЛОЖЕНИЯ

Приложение Д – Код программы “Пчелиного алгоритм”

Приложение Д

Код программы Пчелиного алгоритм

Листинг Д.1 — Основной алгоритм программы

```
import numpy as np

def f(x, y):
    return (1.5 - x + x*y)**2 + (2.25 - x + x*(y**2))**2 + (2.625 -
x + x*(y**3))**2

class Bee():
    def __init__(self):
        self.x = 0
        self.y = 0
        self.f = 0

    def calc(self):
        self.f = f(self.x, self.y)
        return self.f

    def calc_x_area(self, inaccuracy):
        return self.x - inaccuracy, self.x + inaccuracy

    def calc_y_area(self, inaccuracy):
        return self.y - inaccuracy, self.y + inaccuracy
        break

class Colony():
    def __init__(self):
        self.scouts = 10
        self.best = 5
        self.worst = 2
        self.best_area = 2
        self.worst_area = 1
        self.inaccuracy = 1.5

    def bee_attack(self, iter, minX, maxX, minY, maxY):
        beeZ = []
        for i in range (iter):
            bee = Bee()
            bee.x = np.random.uniform(minX, maxX)
            bee.y = np.random.uniform(minY, maxY)
            bee.x = np.clip(bee.x, -4.5, 4.5)
            bee.y = np.clip(bee.y, -4.5, 4.5)
            bee.calc()
            beeZ.append(bee)
        beeZ.sort(key=lambda b: b.f)
        return beeZ
```

Листинг Д.2 — Продолжение листинга Д.1

```
def find_best(self, iter):
    beeZ = self.bee_atack(self.scouts, -4.5, 4.5, -4.5, 4.5)
    beeZ = beeZ[:self.best_area + self.worst_area]

    cur_beeZ = []
    for i in range(iter):
        for bee in range(self.best_area):
            minX, maxX = beeZ[bee].calc_x_area(self.inaccuracy)
            minY, maxY = beeZ[bee].calc_y_area(self.inaccuracy)
            cur_beeZ += self.bee_atack(self.best, minX, maxX,
minY, maxY)

            for bee in range(self.worst_area):
                minX, maxX = beeZ[self.best_area +
bee].calc_x_area(self.inaccuracy)
                minY, maxY = beeZ[self.best_area +
bee].calc_y_area(self.inaccuracy)
                cur_beeZ += self.bee_atack(self.worst, minX, maxX,
minY, maxY)

        cur_beeZ.sort(key=lambda b: b.f)
        cur_beeZ = cur_beeZ[:self.best_area + self.worst_area]

    return cur_beeZ[0]

if __name__ == '__main__':
    print("Эталон: f(3, 0.5) = 0")

    c = Colony()
    iter = 1000

    best_bee = c.find_best(iter)
    print(f"Результат x = {best_bee.x:.5f} y = {best_bee.y:.5f} f =
{best_bee.f:.5f}")
```